**MTech I – ACC Lab: 2017-18**

**Assignment 4: Divide and Conquer**

- For a given problem, analyze the *Divide and Conquer*, design an algorithm and implement the same based on appropriate algorithm design technique.

- For each of the problems the Divide and Conquer needs to be compared with the Brute Force technique. Further, compare the time complexity for both of them for the problem assigned to you.

- Along with the implementations, for submissions, prepare a report and a presentation on the solution of the problem given to you. (The basic format of the report is uploaded on moodle).

|  |  |  |  |
| --- | --- | --- | --- |
| **Group No.** | **Name** | **Roll No.** | **Problem No** |
| **1** | Anmol Verma | P18CO006 | **2** |
| Ankur Bang | D18CO001 |
|  |  |  |  |
| **2** | Jyoti Gage | P18CO003 | **3** |
| Harshita Rao | P18CO010 |
|  |  |
|  |  |  |  |
| **3** | Dilip Puri | P18CO008 | **4** |
| Akash Banchhor | P18CO011 |
| Nishant Singh Chauhan | P18CO012 |
|  |  |  |  |
| **4** | Shivani Makwana | P18CO005 | **1** |
| Alice Anjali | P18CO007 |
| Sheetal Bairangi | P18CO015 |
|  |  |  |  |
| **5** | Ashish Tripathi | P18CO001 | **5** |
| Mochi Kishan | P18CO009 |
| Pratik Patel | P18CO016 |
|  |  |  |  |
| **6** | Ashtankar Akshay | P18CO004 | **6** |
| Tejas Bavaskar | P18CO013 |
| Gopal | P18CO014 |

|  |  |
| --- | --- |
| **Problem No** | **Problem** |
| 1 | You are interested in analyzing some hard-to-obtain data from two separate databases. Each database contains n numerical values - so there are 2n values total - and you may assume that no two values are the same. You would like to determine the median of this set of 2n values, which we will define here to be the nth smallest value.  However the only way you can access these values is through queries to the database. In a single query, you can specify a value k to one of the two database, and the chosen database will return kth smallest value that it contains. Since queries are expensive, you would like to compute the median using as few queries as possible.  Give an algorithm that finds the median value using at most O(logn) queries. |
| 2 | Recall the problem of finding the number of inversions. As in the text, we are given a sequence of n numbers a1, a2, …, an which we assume are all distinct, and we define an inversion to be pair i < j such that ai > aj.  We motivated the problem of counting inversions as a good measure of how different two orderings are. However, one might feel that this measure is too sensitive. Let’s call a pair a significant inversion if i < j and ai > 2 \* aj. Give an O(nlogn) algorithm to count the number of significant inversions between two orderings. |
| 3 | Suppose you’re consulting for a bank that’s concerned about fraud detection, and they come to you with the following problem. They have a collection of n bank cards that they’ve confiscated, suspecting them of being used in fraud. Each bank card is a small plastic object, containing a magnetic stripe with some encrypted data, and it corresponds to a unique account in the bank. Each account can have many bank cards corresponding to it, and we’ll say that two bank cards are equivalent if they correspond to the same account. It’s very difficult to read the account number off a bank card directly, but the bank has a high-tech "equivalence tester" that takes two bank cards and, after performing some computations, determines whether they are equivalent.  Their question is the following: among the collection of n cards, is there a set of more than n/2 of them that are all equivalent to one another? Assume that the only feasible operations you can do with the cards are to pick two of them and plug them in to the equivalence tester. Show how to decide the answer to their question with only O (n log n) invocations of the equivalence tester. |
| 4 | Hidden surface removal is a problem in computer graphics that scarcely needs an introduction: When Woody is standing in front of Buzz, you should be able to see Woody but not Buzz; When Buzz is standing in front of Woody .... well, you get the idea.  The magic of hidden surface removal is that you-can often compute things faster than your intuition suggests. Here’s a clean geometric example to illustrate a basic speed-up that can be achieved. You are given n non vertical lines in the plane, labeled L1, L2, …, Ln with the ith line specified by the equation y = aix + bi. We will make the assumption that no three lines all meet at a single point. We say line Li is uppermost at a given x-coordinate xo if its y coordinate at xo is greater that the y coordinates of all the other lines at xo: aixo + bi > ajxo + bj for all j ≠ i. We say line Li is visible if there is some x coordinates at which it is uppermost intuitively, some portion of it can be seen if you look down from y = infinity.  Give an algorithm that takes n lines as input and in O(nlogn) time returns all of the ones that are visible. |
| 5 | Consider an n-node complete binary tree T, where n = 2d – 1 for some d. Each node v of T is labeled with a real number xv. You may assume that the real numbers labelling the nodes are all distinct. A node v of T is a local minimum if the label xv is less that the label xw for all nodes w that are joined to v by an edge.  You are given such a complete binary tree T, but the labeling is only specified in the following implicit way: for each node v, you can determine the value xv by probing the node v. Show how to find a local minimum of T using only O(logn) probes to the nodes of T. |
| 6 | Suppose you are given an n \* n grid graph G. (An n \* n grid graph is just the adjacency graph of n \* n chessboard. To be completely precise, it is a graph whose node set is the set of all ordered pairs of natural numbers (i, j), where 1 ≤ i ≤n and 1 ≤ j ≤n; the nodes (i, j) and (k, l) are joined by an edge if and only if |i-k| + |j-l| = 1.)  Each node v is labeled by a real number xv; you may assume that all these labels are distinct. Show how to find a local minimum of G using only O(n) probes to the nodes of G. (Note that G has n2 nodes.) |